WHAT IS CLAIMED IS:

1. A computer-implemented method for conducting a negotiation comprising an exchange of messages between first and second entities, the method comprising:

   providing a finite state machine associated with the first entity, the finite state machine having a plurality of states;

   maintaining the finite state machine in one of its states matching a stage of a negotiation between the first and second entities;

   at the first entity conducting a negotiation with the second entity by exchanging messages with the second entity, each of the messages comprising an external aspect containing information determined by a current state of the finite state machine and an internal aspect.

2. The method of claim 1 comprising receiving at the first entity a message from the second entity, the message comprising an external aspect and an internal aspect and providing the external aspect of the message as input to the finite state machine.

3. The method of claim 2 comprising providing the internal aspect of the message to a checker, receiving a result code from the checker and combining the result code with the external aspect of the second message before providing the external aspect of the message as input to the finite state machine.

4. The method of claim 3 wherein the external aspect of the message comprises an integer and the result code comprises zero or a negative integer.

5. The method of claim 4 wherein the finite state machine has a set of transition functions:
$\delta(q, m) = m+1$ if $0 \leq m \leq q+1$ and $m \leq 2n$;
$\delta(q, m) = q$ otherwise;
$\lambda(q, m) = m+1$ if $1 \leq m \leq q+1$ and $m \leq 2n$; and,
$\lambda(q, m) = \epsilon$ otherwise
with $Q = \Sigma = \Delta = \{0, 1, \ldots, 2n+1\}$ where Q is a finite set of states, q_0 is an initial one of the Q states, Σ is a finite input alphabet, Δ is an output alphabet; $\delta: Q \times \Sigma \rightarrow Q$ is a transition function, λ is a mapping from $Q \times \Sigma$ to Δ.

6. The method of claim 5 wherein the set of transition functions comprises:
$\delta(q_0, \epsilon) = q_0$; and,
$\lambda(q_0, \epsilon) = q_0$;
wherein ϵ represents an empty input.

7. The method of claim 6 wherein the set of transition functions comprises:
$\delta(q, \epsilon) = q-2$ if $2 \leq q$; and
$\lambda(q, \epsilon) = q-2$ if $2 \leq q$.

8. The method of claim 7 comprising periodically providing an ϵ input to the finite state machine.

9. The method of claim 4 wherein the finite state machine comprises a set of transition functions and the set of transition functions does not permit the finite state machine to undergo any transitions to any state higher than a next-higher state.

10. The method of claim 9 comprising periodically providing an ϵ input to the finite state machine, and, on a fraction p of the ϵ inputs provided when the finite state machine is not in its initial state, causing the finite state machine to undergo a transition to the initial state.

11. The method of claim 10 comprising after an ϵ input causes the finite state machine to undergo a transition to the initial state, determining whether a negotiation is commenced again with the same second entity and, if so, decreasing p.

12. The method of claim 2 wherein the first entity comprises a service the negotiation relates to configuration of the service, and the internal aspect of the received message comprises one or more parameters related to configuration of the service.

13. The method of claim 11 comprising incorporating the parameters into a configuration object for the service.

14. The method of claim 13 wherein the service comprises a network connectivity service.

15. In a computer system comprising a plurality of entities including first and second entities and one or more data communication channels by way of which the entities can exchange messages with one another, a method by way of which the first entity can obtain a sequence of sets of one or more parameters from the second entity, the method comprising:

a) providing first and second finite state machine at the first and second entities respectively, each of the finite state machines having a plurality of states including an initial state and a final state;

b) setting a current state of the first finite state machine to the initial state;

c) generating a message comprising an external aspect and an internal aspect, the external aspect determined by the current state of the finite state machine, the internal aspect containing information specifying a next set of required parameters; and,

d) sending the message to the second entity.

16. The method of claim 15 comprising subsequently receiving at the first entity a response message from the second entity, the response message comprising an external aspect and an internal aspect, the external aspect determined by a current state of the second finite state machine, the internal aspect comprising the next set of required parameters, and providing the external aspect of the response message as input to the first finite state machine.

17. The method of claim 16 comprising passing the internal aspect of the message to a computational part of the first entity.

18. The method of claim 17 comprising receiving a result code from the computational part of the first entity and combining the result code with the external aspect of the response message before providing the external aspect of the response message as input to the first finite state machine.

19. The method of claim 18 comprising allowing the finite state machine to undergo a transition to a new current state in response

to the provided input and subsequently repeating the steps of generating and sending a message.

20. The method of claim 15 wherein the computer system comprises a computer network, the entities are associated with devices on the computer network.

21. The method of claim 20 comprising, at the first entity, maintaining a cache comprising information identifying other entities on the network wherein, when the current state of the first finite state machine is the initial state, the method comprises identifying another entity on the network as the second entity by making a random selection of the second entity from the cache.

22. The method of claim 21 comprising evaluating a distance function for each entity in the cache wherein making a random selection of the second entity from the cache comprises weighting each entity in the cache by an amount determined by the distance function for that entity.

23. The method of claim 22 wherein the distance function for another entity is based upon a time taken for a message to be exchanged with the other entity.

24. The method of claim 21 comprising checking for starvation of an entity type on the network by checking the cache of an entity on the network for information identifying entities of the entity type.

25. The method of claim 21 comprising determining a state of each of a plurality of finite state machines on the network and checking for

a deadlock condition on the network based on the determined states.

26. The method of claim 21 comprising checking for a deadlock condition on the network by periodically computing the sum

$$r = \sum_{q} \left\lfloor \frac{q}{2} \right\rfloor$$

over all of the finite state machines on the network wherein ***q*** is an integer representing the current state of a finite state machine and comparing the sum to a threshold value.

27. The method of claim 26 wherein the threshold value is $2nm$ wherein there are m pairs of finite state machines and each of the negotiations can be concluded in n rounds.

28. A networkable device comprising a service and a resource allocation component, the resource allocation component comprising a finite state machine having a plurality of possible states including an initial state and a final state, the resource allocation component configured to make available a resource to the service when the finite state machine is in the final state.

29. The device of claim 28 wherein the device comprises a configuration object for the resource and making the resource available to the service comprises enabling access to the configuration object.

30. The device of claim 29 wherein the finite state machine has a set of transition functions:

$\delta(q, m) = m+1$ if $0 \leq m \leq q+1$ and $m \leq 2n$;

$\delta(\boldsymbol{q}, \boldsymbol{m}) = \boldsymbol{q}$ otherwise;
$\lambda(\boldsymbol{q}, \boldsymbol{m}) = \boldsymbol{m}+1$ if $1 \leq \boldsymbol{m} \leq \boldsymbol{q}+1$ and $\boldsymbol{m} \leq 2\boldsymbol{n}$; and,
$\lambda(\boldsymbol{q}, \boldsymbol{m}) = \epsilon$ otherwise
with $\boldsymbol{Q} = \Sigma = \Delta = \{0, 1, \ldots, 2\boldsymbol{n}+1\}$ where $\boldsymbol{Q}$ is a finite set of states, $\boldsymbol{q_0}$ is an initial one of the $\boldsymbol{Q}$ states, Σ is a finite input alphabet, Δ is an output alphabet; δ: $\boldsymbol{Q} \times \Sigma \rightarrow \boldsymbol{Q}$ is a transition function, λ is a mapping from $\boldsymbol{Q} \times \Sigma$ to Δ.

31. The device of claim 30 wherein the set of transition functions comprises:
$\delta(\boldsymbol{q_0}, \epsilon) = \boldsymbol{q_0}$; and,
$\lambda(\boldsymbol{q_0}, \epsilon) = \boldsymbol{q_0}$;
wherein ϵ represents an empty input.

32. The device of claim 31 wherein the set of transition functions comprises:
$\delta(\boldsymbol{q}, \epsilon) = q-2$ if $2 \leq \boldsymbol{q}$; and
$\lambda(q, \epsilon) = \boldsymbol{q}-2$ if $2 \leq \boldsymbol{q}$.

33. The device of claim 32 comprising a timer connected to periodically provide an ϵ input to the finite state machine.

34. The device of claim 27 wherein the finite state machine has a transition function which does not permit transitions to any state higher than a next-higher state.

35. A resource allocation component for networking a device on a data communication network, the resource allocation component comprising a service cache and a finite state machine corresponding to a service of the device, the finite state machine having a plurality of possible states including an initial state and a

final state, the resource allocation component moving the finite state machine to another state upon receiving a message from a corresponding finite state machine and moving the finite state machine to a final state upon receiving a message confirming the availability of a resource needed by the service.